Application Note 130

VTOS DDR: QorIQ P4/P5 Processor Addendum

February 17, 2015

Table of Contents

Overview *3*

Loading the VTOS DDR image onto your board *3*

Using Freescale CodeWarrior Development Studio 3

Manually Configuring CPC Memory 12

Loading VTOS DDR firmware images 14

DDR Board Files *16*

Timing Based Files 16

Register Based Files 16

DDR Clock Tuning *18*

Executing the DDR Clock Tuning 18

Interpreting the tuning results 20

Additional Contact Information *22*

About Kozio, Inc. *22*

© Copyright Kozio, Inc. 2014

Overview

This addendum describes the VTOS DDR features specific to the QorIQ P4 and QorIQ P5 processor families, manufactured by Freescale.

The QorIQ P4 processor is based on the e500mc microprocessor core and runs in 32-bit mode only. The e500mc microprocessor supports a 36-bit physical address space.

The QorIQ P5 processor is based on the e5500 microprocessor core, capable of running in 32-bit mode or 64-bit mode. The e5500 microprocessor supports a 36-bit physical address space. The VTOS DDR firmware image only supports 32-bit operation of the e5500 microprocessor core.

For both microprocessor core types, the VTOS DDR firmware enables the Memory Management Unit (MMU) and supports testing of the entire 36-bit address space (up to 64 GiB of memory).

Loading the VTOS DDR image onto your board

The VTOS DDR image is configured to run directly from the CoreNet platform cache (CPC) memory of the QorIQ P4 and QorIQ P5. The default interface used for communication is through the UART1_SOUT and UART1_SIN pins on the processor.

VTOS DDR Image Information: QorIQ P4/P5	
Load Address	0xFFF00000
Entry Point	0xFFF04000 (also specified by the ELF file)
CCSR Base	0xF_FE000000 physical address 0xFE000000 virtual address

Using Freescale CodeWarrior Development Studio

The VTOS DDR installation includes example projects for CodeWarrior Development Studio that you use to configure the CPC memory, load the VTOS DDR firmware image, and start the processor execution.

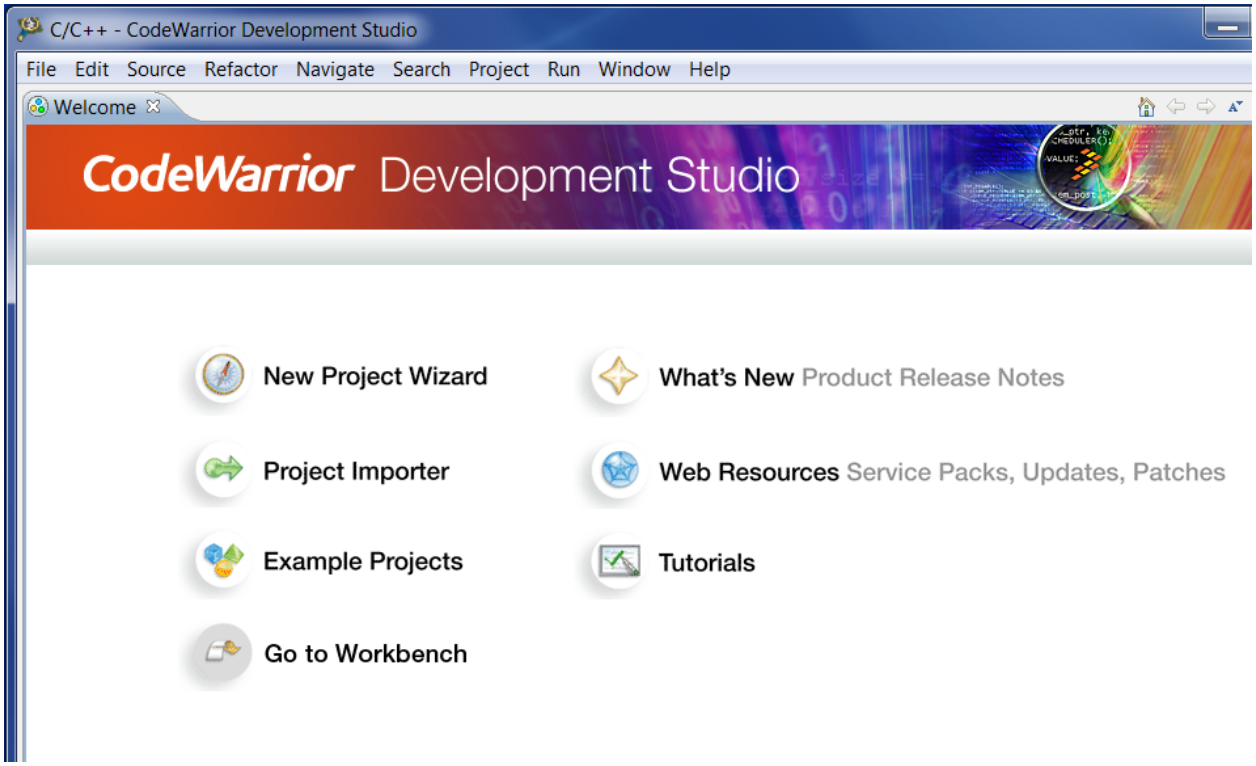
The example projects are located under the default installation directory `%KOZIO_VTOS_DDR_HOME%\etc`.

CodeWarrior Development Studio Walkthrough

1. Download and install CodeWarrior Development Studio for Power Architecture from Freescale. Note that you will need to purchase a license from Freescale in order to run CodeWarrior Development Studio past the evaluation period.
2. Launch CodeWarrior Development Studio.

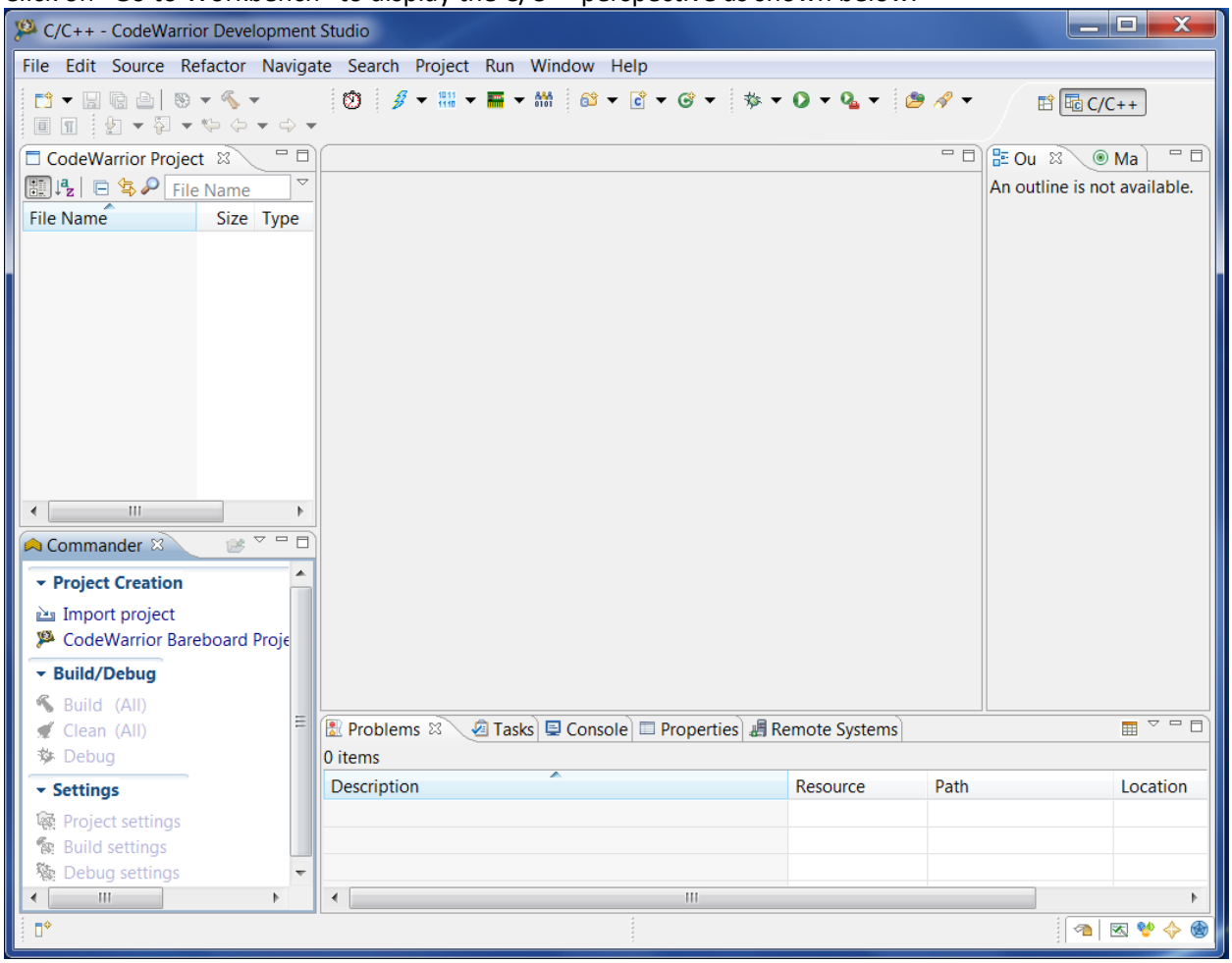
© Copyright Kozio, Inc. 2014

a. If this is the first time you have started CodeWarrior Development Studio, you will be presented with the Welcome Screen as shown below.

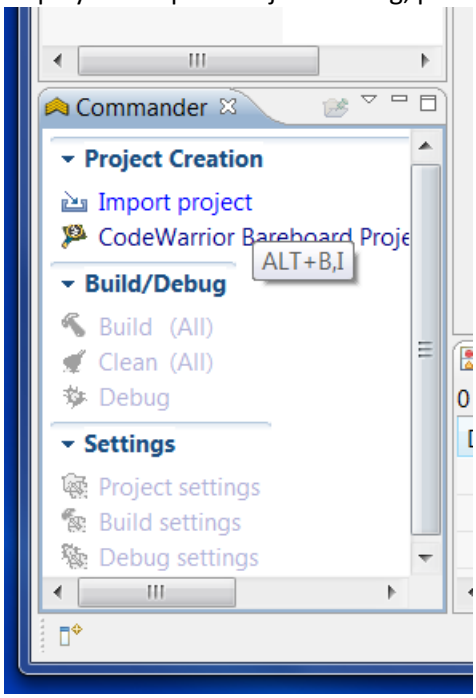


© Copyright Kozio, Inc. 2014

b. Click on “Go to Workbench” to display the C/C++ perspective as shown below.

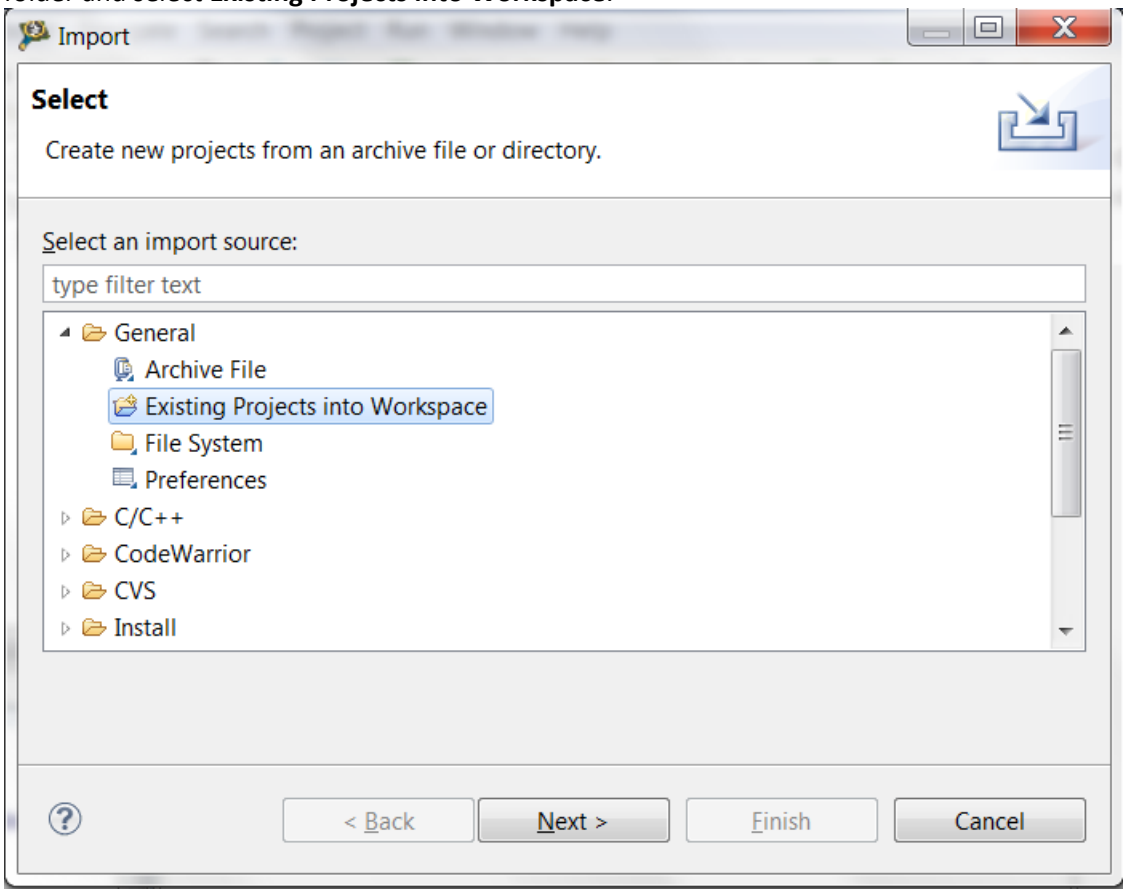


3. Import a VTOS DDR download project. This can be done in one of two ways.
   a. Method 1: Click on the “Import project” icon in the Command window as shown below to display the Import Projects dialog, proceed to step 4.



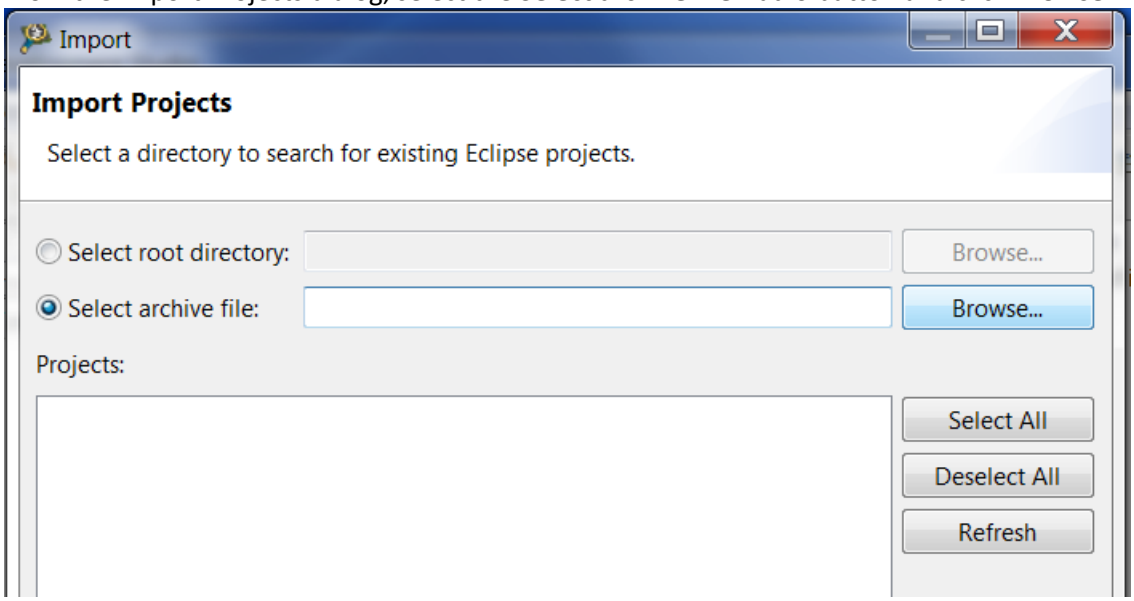
© Copyright Kozio, Inc. 2014

b. Method 2: Select **File->Import Project…** to bring up the Import dialog. Expand the **General** folder and select **Existing Projects into Workspace**.



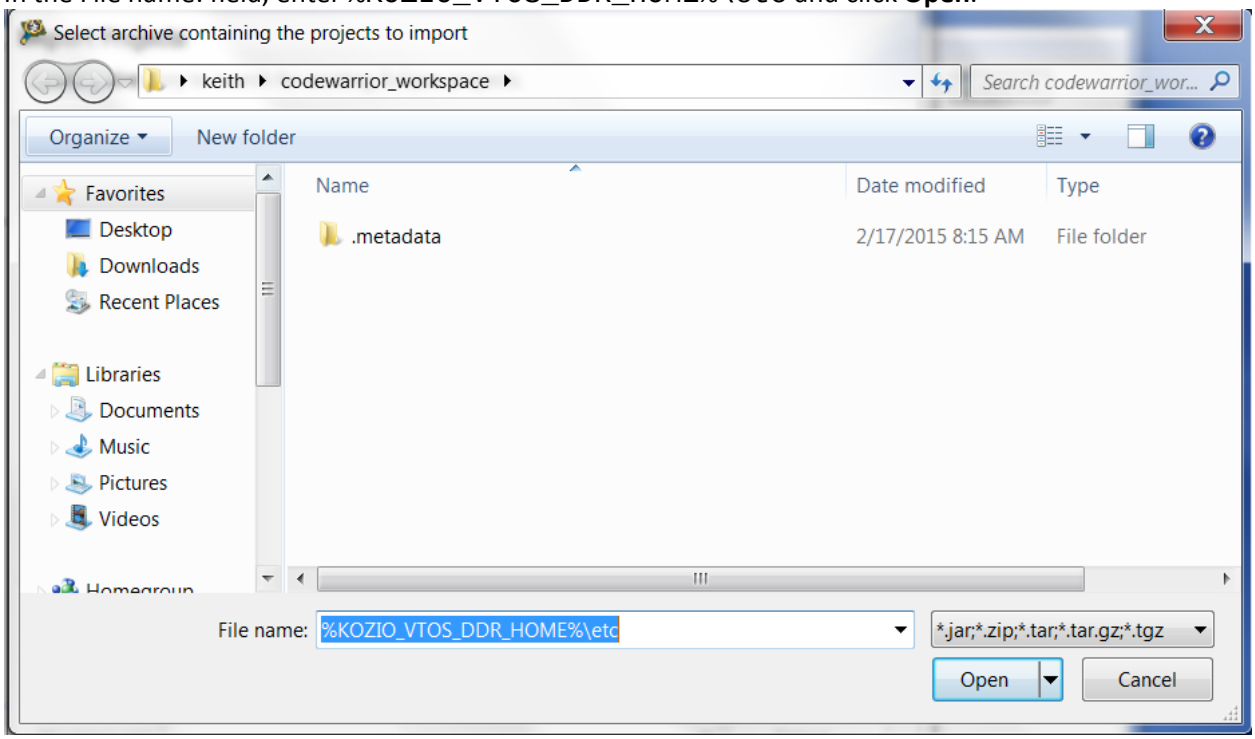
Click **Next>** to display the Import Projects dialog.

4. Import the VTOS DDR project:
   a. From the Import Projects dialog, select the **Select archive file:** radio button and click **Browse**.

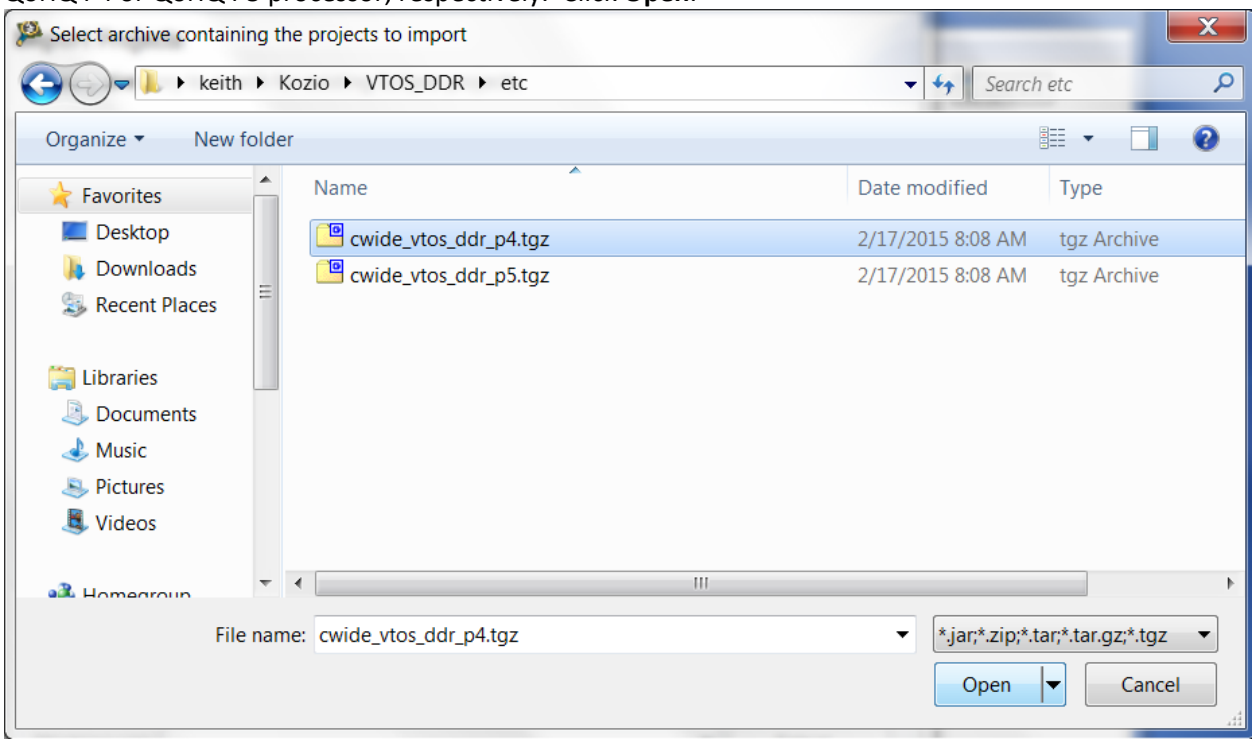


© Copyright Kozio, Inc. 2014

b. In the File name: field, enter `%KOZIO_VTOS_DDR_HOME%\etc` and click **Open**.

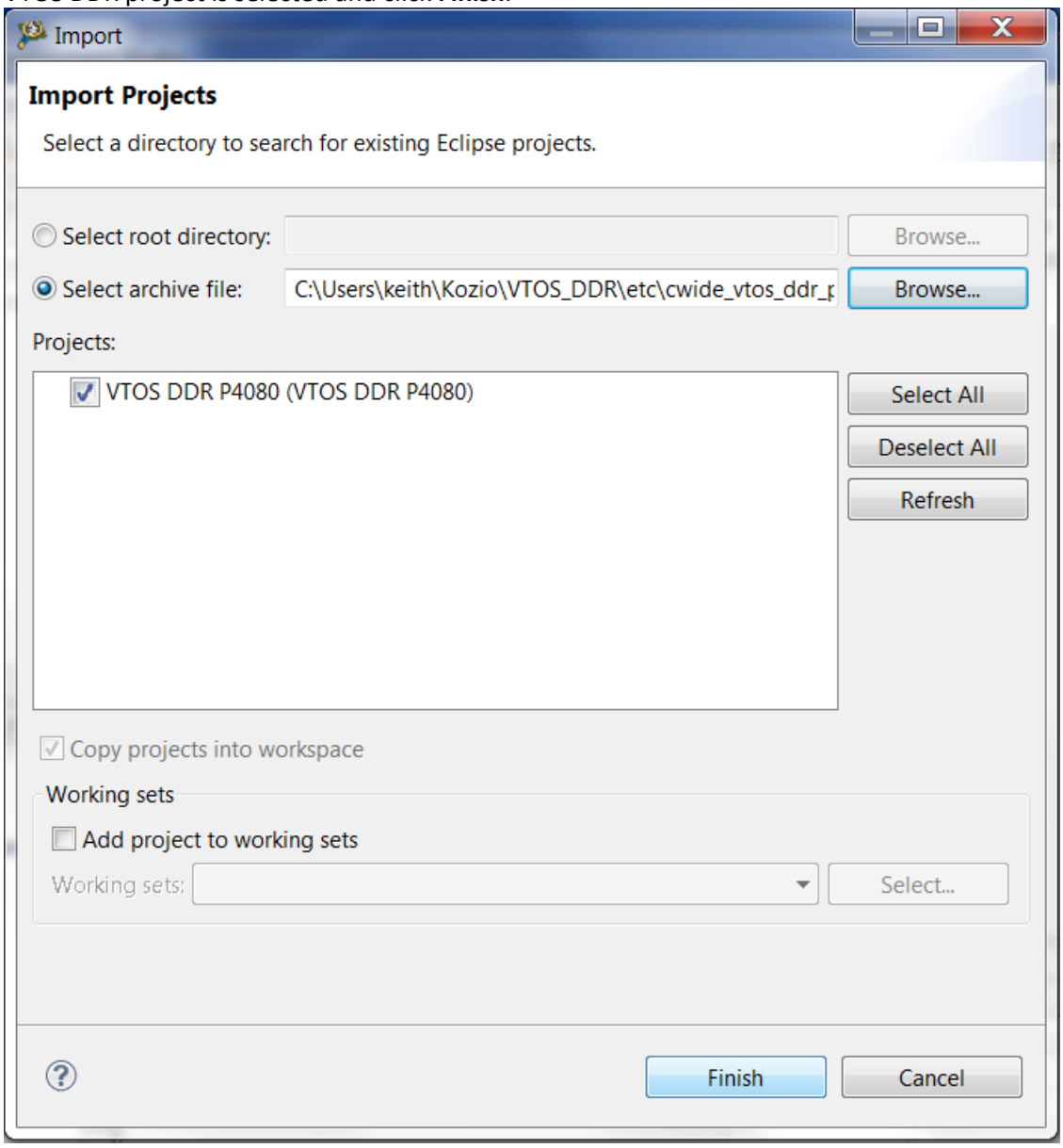


c. Now select the correct CodeWarrior project archive for the processor family you are using. Select the file `cwide_vtos_ddr_p4.tgz` or `cwide_vtos_ddr_p5.tgz` when using a QorIQ P4 or QorIQ P5 processor, respectively. Click **Open**.



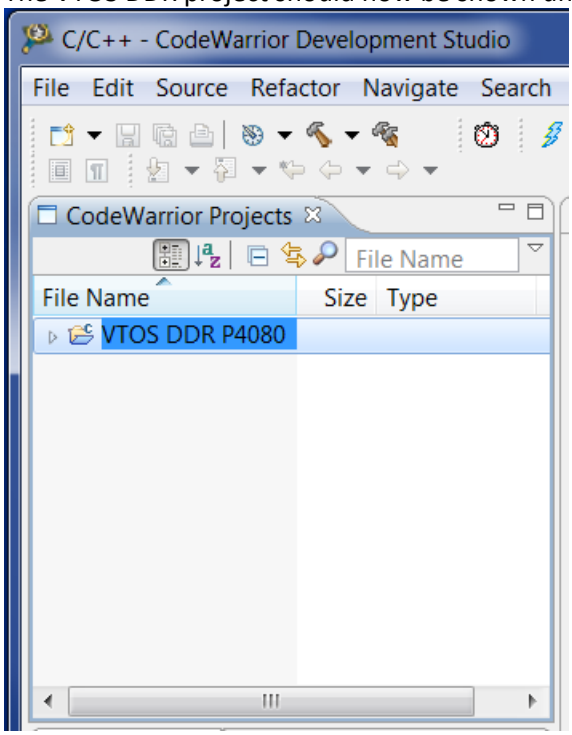
© Copyright Kozio, Inc. 2014

d. The Import dialog should now show the VTOS DDR project under the Projects view. Ensure the VTOS DDR project is selected and click **Finish**.

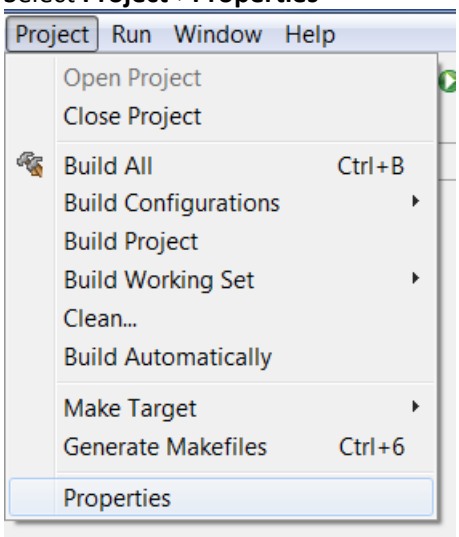


© Copyright Kozio, Inc. 2014

e. The VTOS DDR project should now be shown under the **CodeWarrior Projects** tab.

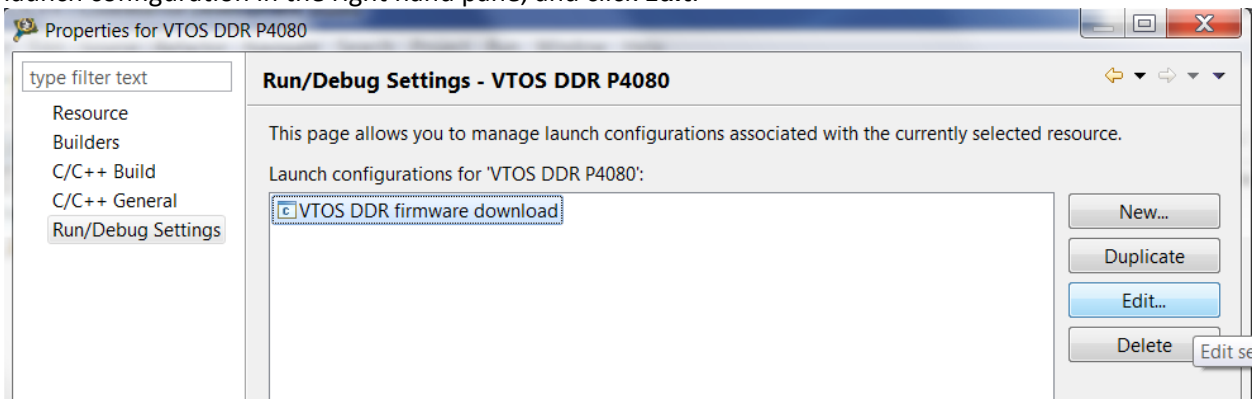


5. The final configuration step is to verify the VTOS DDR Debug/Run settings. By default, the project is configured as follows:
   If your CodeWarrior hardware and board settings match those listed above, proceed to step 7.
   a. Connection type: USB TAP
   b. C/C++ download application: VTOS DDR firmware image for boards using a 100 MHz SYSCLK (vtos.qoriq_p4_ddr.clkin_100.elf or vtos.qoriq_p5_ddr.clkin_100.elf)
6. Optional step: changing the VTOS DDR firmware image or changing the connection type.
   a. Select **Project->Properties**

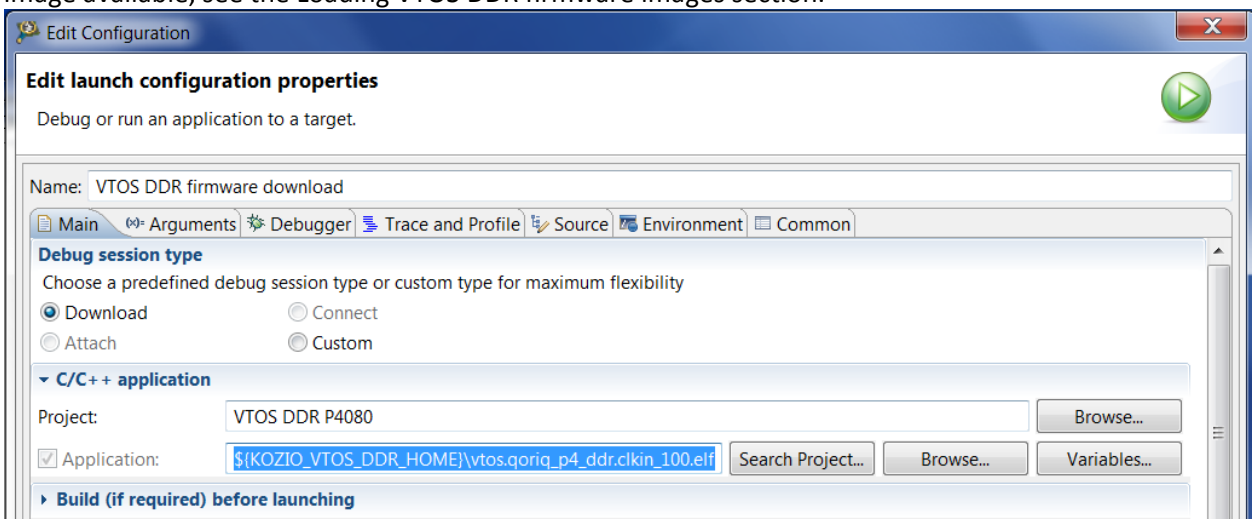


© Copyright Kozio, Inc. 2014

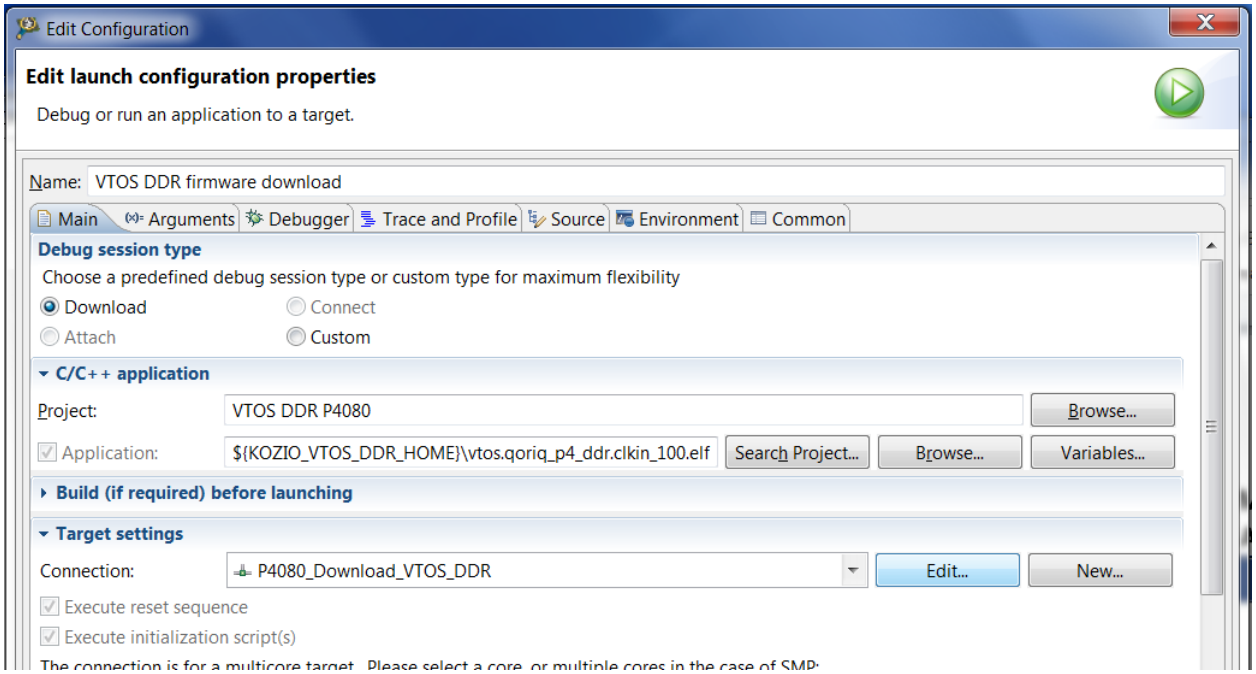
b. Select **Run/Debug Settings** in the left hand pane, select the **VTOS DDR firmware download** launch configuration in the right hand pane, and click **Edit**.



c. To change the VTOS DDR firmware file, select the **Main** tab and modify the C/C++ application field to the desired VTOS DDR firmware file. Click **Apply**. For more details on the firmware image available, see the Loading VTOS DDR firmware images section.

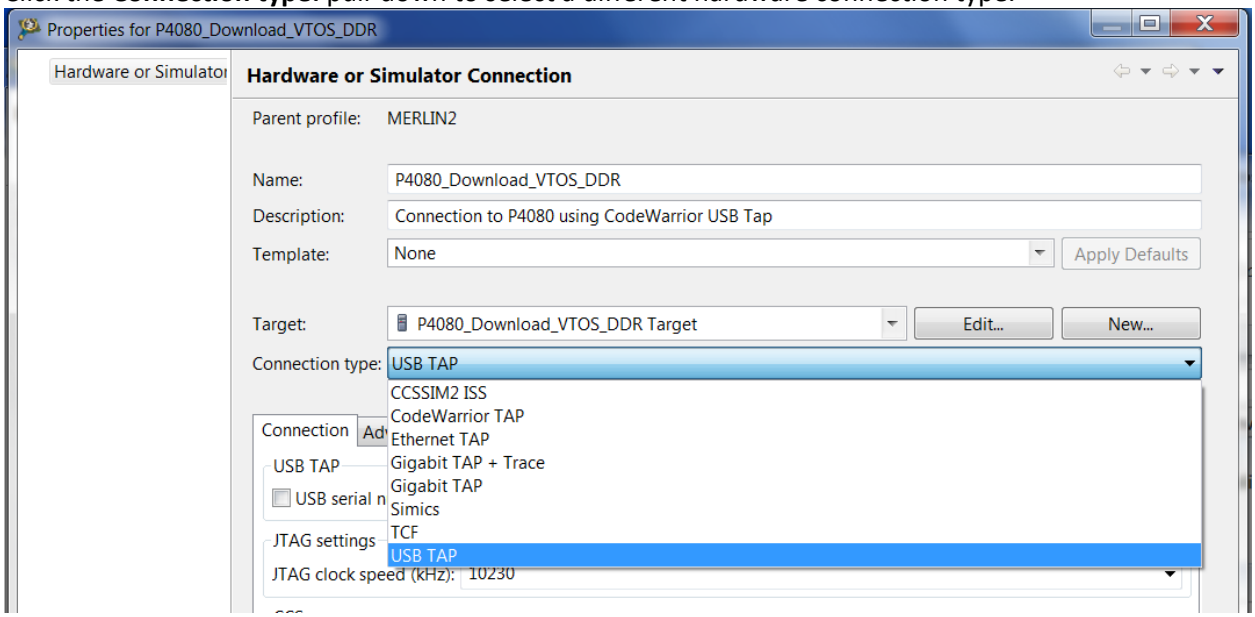


d. To change the CodeWarrior connection type, select the **Main** tab and under **Target Settings** select **Edit…**

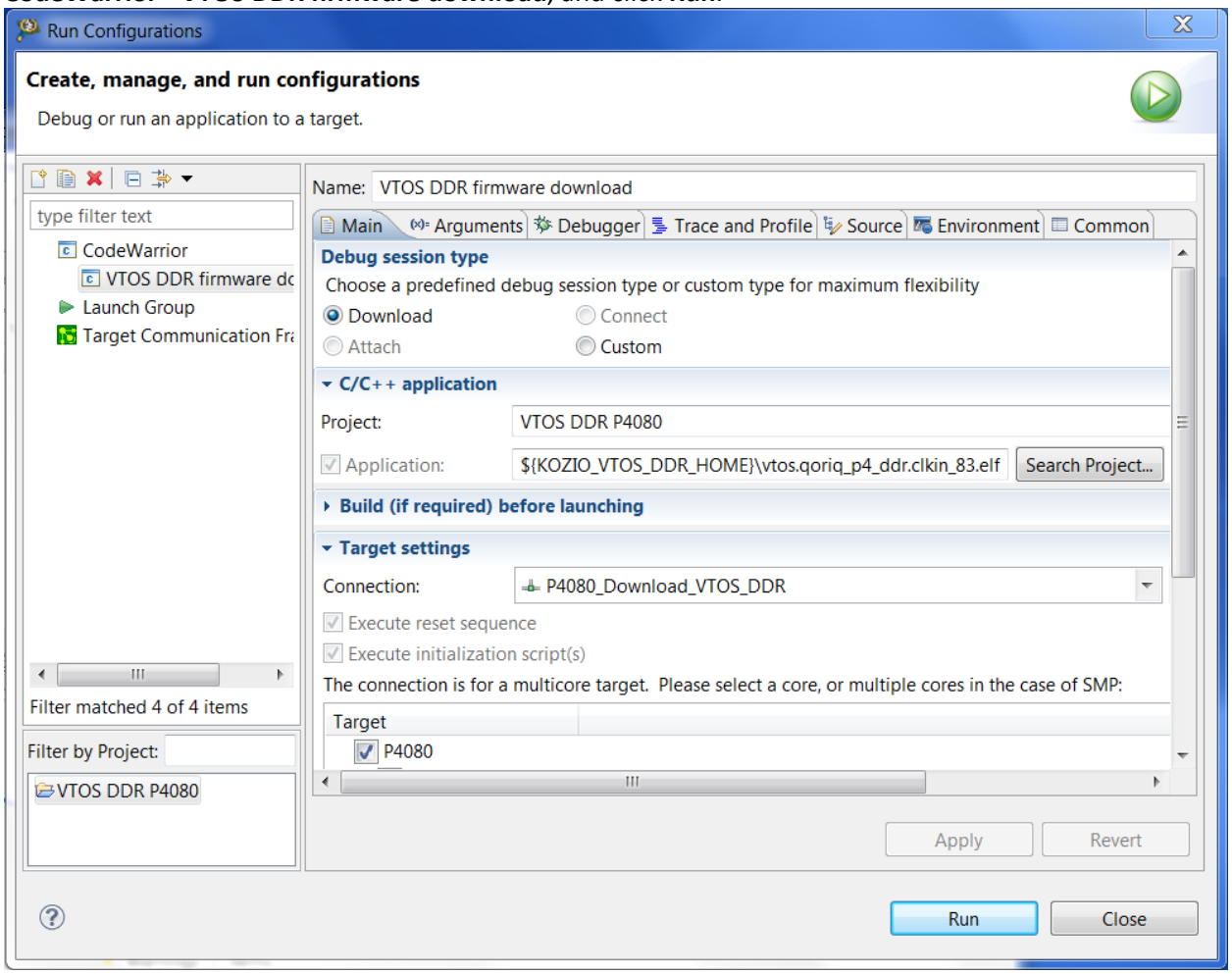


© Copyright Kozio, Inc. 2014

e. Click the **Connection type:** pull-down to select a different hardware connection type.



7. To launch the VTOS DDR firmware image, select **Run->Run Configurations**. In the left-hand pane select **CodeWarrior->VTOS DDR firmware download**, and click **Run**.



8. Confirm that the VTOS DDR firmware image started. Launch the VTOS DDR application and

© Copyright Kozio, Inc. 2014

Manually Configuring CPC Memory

When the QorIQ P4 or P5 processor is brought out of reset the processor operates in 32-bit mode and only the last 4 KiB of the 32-bit address map is accessible (address range 0xFFFF_F000 to 0xFFFF_FFFF). This means that out of reset, the CCSR (Configuration, Control, and Status Register) register space is not accessible. In addition, the CoreNet Platform Cache is disabled and must be configured as SRAM before attempting to load the VTOS DDR firmware image onto a QorIQ P4 or QorIQ P5.

Follow the instructions below to enable CCSR register space access and to configure the CPC (CoreNet Platform Cache) memory as SRAM. The VTOS DDR installer provides an example JTAG configuration files for the BDI3000 debugger from Abatron under the `%KOZIO_VTOS_DDR_HOME%\etc\` directory.

1. Create a TLB1 entry to enable access to the CCSR (Configuration, Control, and Status Register) space. The physical address of the CCSR space from reset is 0x0_FE00_0000.

		CCSR TLB1 Initial Setting: translate 0xFE00_0000 to 0x0_FE00_0000
Register	Settings	Field Description
MAS0	0x10000000	TLBSEL=1b: select TLB1
		ESEL=0x00: select TLB1 entry 0
MAS1	0x80000500	V = 1b: TLB entry is valid
		TID = 0x00: match all process IDs
		TS = 0:
		TSIZE = 0x5: 1 MiB
MAS2	0xFE00000A	EPN = 0xFE0000: Effective (virtual) address 0xFE000000
		I = 1b: Caching-inhibited
		M = 0b: memory coherency not required
		G = 1b: Guarded memory
		E = 0b: big-endian byte order
MAS3	0xFE00003F	RPN = 0xFE0000: lower 32-bits of real address 0xFE000000
		Permissions = 0x3F: read/write/execute permission for user and supervisor
MAS7	0x00000000	RPN = 0x0: upper 4-bits of real address
MAS4	0x00000000	Default value
MAS5	0x00000000	Default value
MAS6	0x00000000	Default value
MAS8	0x00000000	Default value

2. Write the DCFG_BRR register with a value of 0x00000001 to release core 0 from reset.
   Write DCFG_BRR(0xFE0E00E4) = 0x00000001;
3. Configure the CPC as SRAM by following this sequence. This configures the CPC as SRAM for physical address 0x8_0000_0000, leaving space for up to 32 GiB of DDR memory.

Action	Description
Write CPC1_CPCCSR0(0xFE010000) = 0x00200400	Flash invalidate the CPC and clear all locks.
Delay 500 microseconds to allow flash invalidate to complete	
Write CPC1_CPCHDBCR0(0xFE010F00) = 0x081E0000	Disable speculative read requests.
Write CPC1_CPCSRCR1(0xFE010100) = 0x00000008	SRBARU=0x4: Set the upper 4-bits of the SRAM base address to 0x8
Write CPC1_CPCSRCR0(0xFE010104) = 0x0000000B	SRBARL=0x0000: set lower 32-bits of SRAM base address to 0x00000000 SRAMSZ=00101b: SRAM size 1 MiB

© Copyright Kozio, Inc. 2014

SRAMEN=1b: SRAM mode enable

4. Move the CCSR base to physical address 0xF_FE00_0000.

Action	Description
Write LCC_CCSRBARH(0xFE000000) = 0x0000000F	BASE_ADDR_HIGH = 0xF
Write LCC_CCSRBARL(0xFE000004) = 0xFE000000	BASE_ADDR_LOW = 0xFE000000
Write LCC_CCSRAR(0xFE000008)= 0x80000000	Commit = 1

5. Modify the TLB1 entry created in step 1 to change the physical address of the CCSR to 0xF_FE00_0000.

		CCSR TLB1 Final Setting: translate 0xFE00_0000 to 0xF_FE00_0000
Register	Settings	Field Description
MAS0	0x10000000	TLBSEL=1b: select TLB1 ESEL=0x00: select TLB1 entry 0
MAS1	0x80000700	V = 1b: TLB entry is valid TID = 0x00: match all process IDs TS = 0: TSIZE = 0x7: 16 MiB
MAS2	0xFE00000A	EPN = 0xFE0000: Effective (virtual) address 0xFE000000 I = 1b: Caching-inhibited M = 0b: memory coherency not required G = 1b: Guarded memory E = 0b: big-endian byte order
MAS3	0xFE00003F	RPN = 0xFE0000: lower 32-bits of real address 0xFE000000 Permissions = 0x3F: read/write/execute permission for user and supervisor
MAS7	0x0000000F	RPN = 0xF: upper 4-bits of real address
MAS4	0x00000000	Default value
MAS5	0x00000000	Default value
MAS6	0x00000000	Default value
MAS8	0x00000000	Default value

6. Create a TLB1 entry to access SRAM.

		SRAM TLB1 Final Setting: translate 0xFFF0_0000 to 0x8_0000_0000
Register	Settings	Field Description
MAS0	0x10010000	TLBSEL=1b: select TLB1 ESEL=0x01: select TLB1 entry 1
MAS1	0x80000500	V = 1b: TLB entry is valid TID = 0x00: match all process IDs TS = 0: TSIZE = 0x5: 1 MiB
MAS2	0xFFF0000A	EPN = 0xFFF000: Effective (virtual address) 0xFFF00000 I = 1b: Caching-inhibited M = 0b: memory coherency not required G = 1b: Guarded memory E = 0b: big-endian byte order
MAS3	0x0000003F	RPN = 0x000000: lower 32-bits of real address 0x00000000 Permissions = 0x3F: read/write/execute permission for user and supervisor
MAS7	0x00000008	RPN = 0x8: upper 4-bits of real address

© Copyright Kozio, Inc. 2014

MAS4	0x00000000	Default value
MAS5	0x00000000	Default value
MAS6	0x00000000	Default value
MAS8	0x00000000	Default value

7. Create a local access window (LAW) for SRAM,

Action	Description
Write LAW_LAWBARH0(0xFE000C00) = 0x00000008	BASE_ADDR_HIGH=0x8
Write LAW_LAWBARL0 (0xFE000C04) = 0x00000000	BASE_ADDR_LOW=0x00000000
Write LAW_LAWAR0 (0xFE010100) = 0x81000013	EN=1b: enable local access window TRGT_ID=0x10: target identifier memory complex 1 CSC_ID=0x00 SIZE=0x13: 1 MiB

8. Final step: enable the CPC

Action	Description
Write CPC1_CPCCSR0(0xFE010000) = 0x80000000	CPCE=1: enable the CPC array as cache or memory-mapped SRAM

Note that the process for creating TLB1 entries varies for each JTAG debugger. Consult your JTAG debugger documentation for additional information.

You can verify CPC has been correctly configured for SRAM by using your JTAG debugger to read and write memory starting at address 0xFFF0_0000.

Loading VTOS DDR firmware images

After configuring the CPC memory as SRAM on your QorIQ P5 platform, you are ready to load the VTOS DDR firmware image into the CPC memory.

The VTOS DDR installer provides the following files under %KOZIO_VTOS_DDR_HOME% directory for the QorIQ P4 processor family.

Filename	Description
83 MHz SYSCLK Files	
vtos.qoriq_p4_ddr.clkin_83.bin	VTOS DDR Firmware image for P4 boards using an 83 MHz SYSCLK source, raw binary format.
vtos. qoriq_p4_ddr.clkin_83.elf	VTOS DDR Firmware image for P4 boards using an 83 MHz SYSCLK source, ELF object file format.
vtos. qoriq_p4_ddr.clkin_83.srec	VTOS DDR Firmware image for p4 boards using an 83 MHz SYSCLK source, SREC file format.
100 MHz SYSCLK Files	
vtos. qoriq_p4_ddr.clkin_100.bin	VTOS DDR Firmware image for P4 boards using a 100 MHz SYSCLK source, raw binary format.
vtos. qoriq_p4_ddr.clkin_100.elf	VTOS DDR Firmware image for P4 boards using a 100MHz SYSCLK source, ELF object file format.
vtos. qoriq_p4_ddr.clkin_100.srec	VTOS DDR Firmware image for P4 boards using a 100MHz SYSCLK source, SREC file format.

© Copyright Kozio, Inc. 2014

Common Files	
vtos. qoriq_p4_ddr.elf	VTOS DDR Firmware image loaded by the VTOS DDR application. The VTOS DDR application always uses this file, regardless of whether your P4 board uses an 83 MHz CLKIN or a 100 MHz SYSCLK source.

The VTOS DDR installer provides the following files under %KOZIO_VTOS_DDR_HOME% directory for the QorIQ P5 processor family.

Filename	Description
83 MHz SYSCLK Files	
vtos.qoriq_p5_ddr.clkin_83.bin	VTOS DDR Firmware image for P5 boards using an 83 MHz SYSCLK source, raw binary format.
vtos. qoriq_p5_ddr.clkin_83.elf	VTOS DDR Firmware image for P5 boards using an 83 MHz SYSCLK source, ELF object file format.
vtos. qoriq_p5_ddr.clkin_83.srec	VTOS DDR Firmware image for P5 boards using an 83 MHz SYSCLK source, SREC file format.
100 MHz SYSCLK Files	
vtos. qoriq_p5_ddr.clkin_100.bin	VTOS DDR Firmware image for P5 boards using a 100 MHz SYSCLK source, raw binary format.
vtos. qoriq_p5_ddr.clkin_100.elf	VTOS DDR Firmware image for P5 boards using a 100MHz SYSCLK source, ELF object file format.
vtos. qoriq_p5_ddr.clkin_100.srec	VTOS DDR Firmware image for P5 boards using a 100MHz SYSCLK source, SREC file format.
Common Files	
vtos. qoriq_p5_ddr.elf	VTOS DDR Firmware image loaded by the VTOS DDR application. The VTOS DDR application always uses this file, regardless of whether your P5 board uses an 83 MHz CLKIN or a 100 MHz SYSCLK source.

Select the VTOS DDR firmware image that matches processor type and the SYSCLK setting for your board.

The general process for loading and executing the VTOS DDR firmware image is as follows:

1. Configure CPC memory as SRAM as described in the previous section
2. Load the VTOS DDR firmware image into SRAM. The load address is 0xFFF0_0000 if using a raw binary file.
3. Change the program counter (PC) register to the VTOS DDR firmware entry point 0xFFF0_4000. This step is optional when loading ELF or SREC files.
4. Resume operation of the processor.

It is recommended that use your JTAG debugger to load either an ELF file or an SREC file. Both of these file types provide load address information and start address information. Consult your JTAG debugger documentation for additional information about loading ELF and SREC file types.

© Copyright Kozio, Inc. 2014

DDR Board Files

Please see the VTOS DDR User manual for a detailed description on how to create and use board files. This document details the parameters that are specific to the QorIQ P4/P5 processor family.

> **Tip:** When first determining valid DDR settings for a board design, it is recommended that you disable ECC and the data initialization features of the DDR controller. If the DDR settings are not stable, enabling ECC or the data initialization features can cause the QorIQ P4/P5 processor to hang. Once you have settings that pass all DDR tests with ECC disabled, then you can enable ECC and re-run the DDR configuration and tests.

Timing Based Files

In the current release of VTOS DDR for the QorIQ P4/P5 family, Timing Based board files are not supported. You must use a register based board file.

Register Based Files

Use a Register Based DDR board file if you want to explicitly specify all register values programmed into the DDR controllers of the QorIQ P4/P5 processor.

VTOS DDR supports configuring the DDR1 controller, the DDR2 controller, or configuring both DDR controllers in the QorIQ P4/P5 processor. When both DDR controllers are enabled, VTOS DDR automatically detects whether controller interleaving is enabled and configures the Local Access Window to match the interleaving setting.

Two variables control whether a DDR controller is initialized: *$ddr1_enabled* and *$ddr2_enabled*.

VTOS DDR provides two sets of variables to initialize all DDR controller registers. Variables used to initialize the DDR1 controller are prefixed with *$ddr1_* and variables used to initialize the DDR2 controller are prefixed with *$ddr2_*.

The table below lists all parameters required on the QorIQ P5 processor. For all variables listed below, replace "*x*" with either "*1*"or "*2*" to configure the DDR1 and DDR2 controllers, respectively.

Parameter	Description
$ddr1_enabled	Set to TRUE if your board connects DDR memory chips to the DDR1 memory controller. Set to FALSE if the DDR1 memory controller is not used by your board.
$ddr2_enabled	Set to TRUE if your board connects DDR memory chips to the DDR2 memory controller. Set to FALSE if the DDR2 memory controller is not used by your board.
$ddrx_cs0_bnds	Specifies the value written to the DDRx_CS0_BNDS register.
$ddrx_cs1_bnds	Specifies the value written to the DDRx_CS1_BNDS register.
$ddrx_cs2_bnds	Specifies the value written to the DDRx_CS2_BNDS register.
$ddrx_cs3_bnds	Specifies the value written to the DDRx_CS3_BNDS register.
$ddrx_cs0_config	Specifies the value written to the DDRx_CS0_CONFIG register.
$ddrx_cs1_config	Specifies the value written to the DDRx_CS1_CONFIG register.
$ddrx_cs2_config	Specifies the value written to the DDRx_CS2_CONFIG register.
$ddrx_cs3_config	Specifies the value written to the DDRx_CS3_CONFIG register.
$ddrx_cs0_config_2	Specifies the value written to the DDRx_CS0_CONFIG_2 register.
$ddrx_cs1_config_2	Specifies the value written to the DDRx_CS1_CONFIG_2 register.
$ddrx_cs2_config_2	Specifies the value written to the DDRx_CS2_CONFIG_2 register.
$ddrx_cs3_config_2	Specifies the value written to the DDRx_CS3_CONFIG_2 register.

© Copyright Kozio, Inc. 2014

$ddrx_config	Specifies the value written to the DDRx_DDR_SDRAM_CFG register.
$ddrx_config_2	Specifies the value written to the DDRx_DDR_SDRAM_CFG_2 register.
$ddrx_rcw_1	Specifies the value written to the DDRx_DDR_SRAM_RCW_1 register. Note that this register is only used with registered DIMMs and if the DDR_SRAM_CFG_2[RCW_EN] field is enabled.
$ddrx_rcw_2	Specifies the value written to the DDRx_DDR_SRAM_RCW_2 register. Note that this register is only used with registered DIMMs and if the DDR_SRAM_CFG_2[RCW_EN] field is enabled.
$ddrx_mode	Specifies the value written to the DDRx_DDR_SDRAM_MODE register.
$ddrx_mode_2	Specifies the value written to the DDRx_DDR_SDRAM_MODE_2 register.
$ddrx_mode_3	Specifies the value written to the DDRx_DDR_SDRAM_MODE_3 register.
$ddrx_mode_4	Specifies the value written to the DDRx_DDR_SDRAM_MODE_4 register.
$ddrx_mode_5	Specifies the value written to the DDRx_DDR_SDRAM_MODE_5 register.
$ddrx_mode_6	Specifies the value written to the DDRx_DDR_SDRAM_MODE_6 register.
$ddrx_mode_7	Specifies the value written to the DDRx_DDR_SDRAM_MODE_7 register.
$ddrx_mode_8	Specifies the value written to the DDRx_DDR_SDRAM_MODE_8 register.
$ddrx_interval	Specifies the value written to the DDRx_DDR_SDRAM_INTERVAL register.
$ddrx_clk_cntl	Specifies the value written to the DDRx_DDR_SDRAM_CLK_CNTL register.
$ddrx_wrlvl_cntl	Specifies the value written to the DDRx_DDR_WRLVL_CNTL register.
$ddrx_wrlvl_cntl_2	Specifies the value written to the DDRx_DDR_WRLVL_CNTL_2 register.
$ddrx_wrlvl_cntl_3	Specifies the value written to the DDRx_DDR_WRLVL_CNTL_3 register.
$ddrx_timing_cfg_0	Specifies the value written to the DDRx_TIMING_CFG_0 register.
$ddrx_timing_cfg_1	Specifies the value written to the DDRx_TIMING_CFG_1 register.
$ddrx_timing_cfg_2	Specifies the value written to the DDRx_TIMING_CFG_2 register.
$ddrx_timing_cfg_3	Specifies the value written to the DDRx_TIMING_CFG_3 register.
$ddrx_timing_cfg_4	Specifies the value written to the DDRx_TIMING_CFG_4 register.
$ddrx_timing_cfg_5	Specifies the value written to the DDRx_TIMING_CFG_5 register.
$ddrx_data_init	Specifies the value written to the DDRx_DDR_DATA_INIT register.
$ddrx_zq_cntl	Specifies the value written to the DDRx_DDR_ZQ_CNTL register.
$ddrx_cdr_1	Specifies the value written to the DDRx_DDRCDR_1 register.

Example QorIQ P5 register settings are shown below. In this example, only the DDR2 controller is configured.

```
FALSE -> $ddr1_enabled
TRUE -> $ddr2_enabled

0x000000FF -> $ddr2_cs0_bnds
0x010001FF -> $ddr2_cs1_bnds
0x00000000 -> $ddr2_cs2_bnds
0x00000000 -> $ddr2_cs3_bnds
0x80044402 -> $ddr2_cs0_config
0x80044402 -> $ddr2_cs1_config
0x00000000 -> $ddr2_cs2_config
0x00000000 -> $ddr2_cs3_config
0x00000000 -> $ddr2_cs0_config_2
0x00000000 -> $ddr2_cs1_config_2
0x00000000 -> $ddr2_cs2_config_2
0x00000000 -> $ddr2_cs3_config_2
0xE7040000 -> $ddr2_config
0x24401110 -> $ddr2_config_2
0x00000000 -> $ddr2_rcw_1
0x00000000 -> $ddr2_rcw_2
```

© Copyright Kozio, Inc. 2014

```
0x00441c70 -> $ddr2_mode
0x00180000 -> $ddr2_mode_2
0x00001c70 -> $ddr2_mode_3
0x00180000 -> $ddr2_mode_4
0x00001c70 -> $ddr2_mode_5
0x00180000 -> $ddr2_mode_6
0x00001c70 -> $ddr2_mode_7
0x00180000 -> $ddr2_mode_8
0x18600100 -> $ddr2_interval
0x02800000 -> $ddr2_clk_cntl
0x8675f608 -> $ddr2_wrlvl_cntl
0x00000000 -> $ddr2_wrlvl_cntl_2
0x00000000 -> $ddr2_wrlvl_cntl_3
0x50550104 -> $ddr2_timing_cfg_0
0xbcb58c56 -> $ddr2_timing_cfg_1
0x0fc0c8d8 -> $ddr2_timing_cfg_2
0x010c1000 -> $ddr2_timing_cfg_3
0x00000001 -> $ddr2_timing_cfg_4
0x00001400 -> $ddr2_timing_cfg_5
0xDEADBEEF -> $ddr2_data_init
0x89080600 -> $ddr2_zq_cntl
0x80000000 -> $ddr2_cdr_1
```

Tip: When using a register based board file, it is recommended that you copy one of the reference board register files provided under the directory `%KOZIO_VTOS_DDR_HOME%/boards/qoriq_p5`. Copy and rename an example board file to the directory `%KOZIO_VTOS_DDR_HOME%/boards/user`.

DDR Clock Tuning

The DDR controller provides two parameters that finely control the clocking on the DDR interface: DDRx_DDR_SDRAM_CLK_CNTL[CLK_ADJUST] and DDRx_DDR_WRLVL_CNTL[WRLVL_START].

For a specific board design there is a range of values for both parameters that will operate successfully. Freescale recommends that you identify all valid values for both parameters, and then select the center value to provide the best margins.

VTOS DDR provides the command **`ddr.tune.clocks`** which automatically tries all possible settings for both the CLK_ADJUST and WRLVL_START parameters and then generates a report of the optimal settings found.

Executing the DDR Clock Tuning

Once working settings for the DDR controller on your design are known, run the command **`ddr.tune.clocks`** to find the valid operating window for the CLK_ADJUST and WRLVL_START parameters.

The command can take several minutes to complete. Example output when run on the P5040RDB reference board from Freescale is shown below.

```
kozio> ddr.tune.clocks

------------------------------------------------------------------------
Testing CLK_ADJUST = 5
```

© Copyright Kozio, Inc. 2014

```
-------------------------------------------------------------------
Testing: CLK_ADJUST = 5   WRLVL_START = 8

Configuring DDR3 memory controller
Creating local access window for DDR2
Enabling the DDR2 controller
DRAM data initialization is complete
Memory configuration complete: memory size = 8 GiB

SDRAM: Full burst (64 bit) [00000000_00000000 - 00000000_00ffffff]
............................................................
Passed!
-------------------------------------------------------------------
Testing: CLK_ADJUST = 5   WRLVL_START = 9

Configuring DDR3 memory controller
Creating local access window for DDR2
Enabling the DDR2 controller
DRAM data initialization is complete
Memory configuration complete: memory size = 8 GiB

SDRAM: Full burst (64 bit) [00000000_00000000 - 00000000_00ffffff]
............................................................
Passed!
-------------------------------------------------------------------
Testing: CLK_ADJUST = 5   WRLVL_START = 10

Configuring DDR3 memory controller
Creating local access window for DDR2
Enabling the DDR2 controller
DRAM data initialization is complete
Memory configuration complete: memory size = 8 GiB

SDRAM: Full burst (64 bit) [00000000_00000000 - 00000000_00ffffff]
............................................................
Passed!

// output truncated...

Testing: CLK_ADJUST = 6   WRLVL_START = 13

Configuring DDR3 memory controller
Creating local access window for DDR2
Enabling the DDR2 controller
DRAM data initialization is complete
Memory configuration complete: memory size = 8 GiB

SDRAM: Full burst (64 bit) [00000000_00000000 - 00000000_00ffffff]
...............Bad value at address 0x00000000_00000c80
    expected 0x00000000_00000c80, actual 0x00000200_00000c80
Too many memory errors detected (> 256 errors), ending test
Test failed with error code = 0x00080030

FAILED

// output truncated...

######################################################################
CLK_ADJUST and WRLVL_START tuning completed
######################################################################

                    WRLVL_START
          0    4    8    12   16   20   24   28
CLK_ADJ   ---------------------------------------
  0x00  | 0000 0000 0000 0000 0000 0000 0000 0000
  0x01  | 0000 0000 0000 0000 0000 0000 0000 0000
  0x02  | 0000 0000 0000 0000 0000 0000 0000 0000
```

© Copyright Kozio, Inc. 2014

```
  0x03  | 0001 1111 1100 0000 0000 0000 0000 0000
  0x04  | 0000 0011 1110 0000 0000 0000 0000 0000
  0x05  | 0000 0111 1111 0000 0000 0000 0000 0000
  0x06  | 0000 0000 1111 1000 0000 0000 0000 0000
  0x07  | 0000 0001 1111 1100 0000 0000 0000 0000
  0x08  | 0000 0000 0000 0000 0000 0000 0000 0000

Optimal setting: CLK_ADJUST = 5  WRLVL_START = 8

If using a timing based file, copy and paste the following lines into your board file
    5  -> $ddr.memctrl.clk_adjust
    8  -> $ddr.memctrl.wrlvl_start

If using a register based file, copy and paste the following lines into your board
file
    0x02800000 -> $ddr2_clk_cntl
    0x8675F608 -> $ddr2_wrlvl_cntl
Configuring DDR3 memory controller
Creating local access window for DDR2
Enabling the DDR2 controller
```

Once the DDR clock tuning procedure completes, update your board file with the optimum values reported.

Note that some combinations of the CLK_ADJUST and WRLVL_START parameters can cause a hard hang in the QorIQ_P5 processor. If this happens please reset your board, reload your board file, and try the clock tuning procedure again.

Interpreting the tuning results

Run the command **`tuning.map`** to display the results from the last execution of the DDR clock tuning.

```
kozio> tuning.map
                    WRLVL_START
          0    4    8    12   16   20   24   28
CLK_ADJ   ---------------------------------------
  0x00  | 0000 0000 0000 0000 0000 0000 0000 0000
  0x01  | 0000 0000 0000 0000 0000 0000 0000 0000
  0x02  | 0000 0000 0000 0000 0000 0000 0000 0000
  0x03  | 0000 1111 1100 0000 0000 0000 0000 0000
  0x04  | 0000 0011 1110 0000 0000 0000 0000 0000
  0x05  | 0000 0111 1111 0000 0000 0000 0000 0000
  0x06  | 0000 0000 1111 1000 0000 0000 0000 0000
  0x07  | 0000 0001 1111 1000 0000 0000 0000 0000
  0x08  | 0000 0000 0000 0000 0000 0000 0000 0000

Optimal setting: CLK_ADJUST = 5  WRLVL_START = 8

If using a timing based file, copy and paste the following lines into your board file
    5  -> $ddr.memctrl.clk_adjust
    8  -> $ddr.memctrl.wrlvl_start

If using a register based file, copy and paste the following lines into your board
file
    0x02800000 -> $ddr2_clk_cntl
    0x8675F608 -> $ddr2_wrlvl_cntl
```

Each row of the tuning map corresponds to a single setting of the CLK_ADJUST parameter. At each CLK_ADJUST setting, VTOS DDR tries all possible settings for the WRLVL_START parameter. At each combination of CLK_ADJUST and WRLVL_START, if configuration of the DDR memory controllers was successful and the memory test passed, the map displays a “1”. If configuration failed or the memory test failed, the map displays a “0”.

© Copyright Kozio, Inc. 2014

In the example above, VTOS DDR determined that CLK_ADJUST settings of 3 – 7 had at least one WRLVL_START setting that passed all tests. VTOS DDR then selects CLK_ADJUST setting of 5 as the middle of the CLK_ADJUST window. At the CLK_ADJUST setting of 5 the WRLVL_START operating window is from setting 5 to 11, with the middle of the window at 8.

Even on incomplete runs of the DDR clock tuning, you can retrieve the results from the clock parameter search. In the example below, the DDR clock tuning was prematurely halted by typing CTRL-C in the VTOS DDR command console.

```
kozio> ddr.tune.clocks

------------------------------------------------------------------------
Testing CLK_ADJUST = 5


------------------------------------------------------------------------
Testing: CLK_ADJUST = 5   WRLVL_START = 8

Configuring DDR3 memory controller
Creating local access window for DDR2
Enabling the DDR2 controller
DRAM data initialization is complete
Memory configuration complete: memory size = 8 GiB

SDRAM: Full burst (64 bit) [00000000_00000000 - 00000000_00ffffff]
............................................................
Passed!
------------------------------------------------------------------------
Testing: CLK_ADJUST = 5   WRLVL_START = 9

// output truncated...

------------------------------------------------------------------------
Testing: CLK_ADJUST = 7   WRLVL_START = 12

Configuring DDR3 memory controller

Command aborted, returning to main task...

kozio> tuning.map
                     WRLVL_START
          0    4    8    12   16   20   24   28
CLK_ADJ   ---------------------------------------
  0x00  | 0000 0000 0000 0000 0000 0000 0000 0000
  0x01  | 0000 0000 0000 0000 0000 0000 0000 0000
  0x02  | 0000 0000 0000 0000 0000 0000 0000 0000
  0x03  | 0000 0000 0000 0000 0000 0000 0000 0000
  0x04  | 0000 0000 0000 0000 0000 0000 0000 0000
  0x05  | 0000 0111 1111 0000 0000 0000 0000 0000
  0x06  | 0000 0000 1111 1000 0000 0000 0000 0000
  0x07  | 0000 0000 0000 0000 0000 0000 0000 0000
  0x08  | 0000 0000 0000 0000 0000 0000 0000 0000

Optimal setting: CLK_ADJUST = 5  WRLVL_START = 8

If using a timing based file, copy and paste the following lines into your board file
    5  -> $ddr.memctrl.clk_adjust
    8  -> $ddr.memctrl.wrlvl_start

If using a register based file, copy and paste the following lines into your board
file
    0x02800000 -> $ddr2_clk_cntl
    0x8675F608 -> $ddr2_wrlvl_cntl
```

© Copyright Kozio, Inc. 2014

Additional Contact Information

Kozio, Inc., +1 (303) 776-1356 x1, sales@kozio.com, www.kozio.com
http://www.kozio.com/

About Kozio, Inc.

Kozio, Inc. provides software solutions delivering Embedded Verification IP refined over the years. Kozio's Verification and Test OS (VTOS) is targeted for a specific SoC. VTOS provides a verification platform with deterministic test results, low-level commands for direct hardware access, and built-in system-level tests. It is a light-weight, easily-configurable, bare-metal verification platform. VTOS is a reusable SoC verification solution for RTL simulation, emulation, prototyping, virtualization, and first articles.
For board-level verification and validation, VTOS provides at-speed functional test tools, and high-speed programming tools, used to quickly verify new embedded designs for design correctness and to quickly validate product during production runs. Quickly identify and troubleshoot hardware failures with direct hardware access and a powerful scripting language. Kozio products extend coverage; provide fast booting, fast test execution, and high-speed programming; saving up to 75% of your test program development costs.

© 2014 Kozio, Inc. All rights reserved. Kozio, VTOS, Integration Workbench, and vAccess are trademarks or registered trademarks of Kozio, Inc. All other trademarks and products are the property of their respective owners.